EMCal ToF Calibration and Slewing Effects in Year 5 Cu+Cu, Year 6 p+p and Year 7 Au+Au collisions.

Technical Note

R. Vértesi, G. David, T. Csörgő

June 10, 2008

Abstract

In this note we summarize the methods that were used to produce the photon time of flight calibration constants for the PHENIX EMCal detectors and provide a description and a users manual to the software tools involved in this process. We also improved the reliability of the slewing (walk) correction, which the ToF strongly relies on.

Contents

1 Introduction

The time of flight (ToF, in the units of ns) of a particle to the EMCal detector is reconstructed the following way:

$$\mathrm{ToF} = -t_{\mathrm{BBC0}} - t_0 - \mathrm{LC}(\mathrm{TDC} - \mathrm{walk}) - t_{\mathrm{flash}}$$

where t_0 is the sum of tower t_0 and sector t_0 values for the corresponding tower and run (see Sec. 1.2.1 and Sec. 1.2.2 respectively), LC is the least count (calibration constant for the TDC to ns conversion units), TDC is baseline-subtracted, walk is the correction factor for the ADC dependency of the TDC value (also referred to as the slewing effect, see Sec. 4 for details), $t_{\mathrm{BBC0}}$is the zero time point set by the Beam-Beam Counter and t_{flash} is the ideal photon flight time from the vertex to the detector (ie. the distance in ns). Therefore the ToF of photons should be 0 by definition. Please refer to the PHENIX EMCal documentation site [1] for a more detailed description of the above quantities.

This note is mostly about the multi-step process of determining the walk and the t_0 offsets for photons.

1.1 Slewing (Walk)

Slewing (walk) is the energy-dependent part of the measured photon flighttime. It is usually expressed in terms of TDC(ADC) instead of ToF(E). In order to get the "real" timing, one should correct for the slewing. The slewing correction should be obtained for each individual channel, ie. one should accumulate a high number of calorimeter hits in each tower. The two ways of determining the slewing is to use hits from the reference laser, or photon-like hits from data. See Sec. 4 for a more detailed description.

1.2 Outline of the EMCal ToF calibration

1.2.1 Tower t_0 offsets

First of all, one needs most of other EMCal calibrations ready in the production database, as well as the t_{BBC0} calibration, for all the runs one is going to use. The actual timing values are calculated from raw ADC and TDC using the gain, least count and walk (slewing) constants.

Then, one takes a run with lots of events, and determines a set of timing offsets for each and every tower (25535 in PbSc and PbGl altogether). This is the so called tower-t0 set. This is done by selecting photons hitting a tower, then fitting the peak of the timing distribution.

1.2.2 Sector t_0 Tracing

As the second step of the timing calibration one traces the movement of the sector time peaks (tower-t0 aligned) from run to run: The so called run-by-run sector-t0 offsets.

It is done two ways, with two separate (but similar) packages. In Year 5 Cu+Cu run-by-run sector t0-s were determined offline, from nanoDST files. From Year 6 we tried to go online: After having the tower-t0 set, an online calibration (see Sec. 3) was done for each run right before the raw data files would go to the HPSS. This has the advantage of not needing already produced DST files, and the disadvantage of relying on the early calibrations.

The data flow chart of the discussed processes is on Fig. 1.

2 The Offline Package

This section summarises the capabilities of this package. The original author for the Year 4 version is L. Aphecetche. The code is located in CVS as `offline/analysis/emcTimingOffsets` [2]. All work is done in the Fun4All framework, using the EMCal reconstruction code. There are macros both for working over PRDF eventdata and over nanoDST's and many other utilities are included in the package.

A detailed documentation of all the the classes, variables and methods and a step-by-step installation guide can be found under [3].

For details on the explicit way the package was used on data from different Years, please refer to the later section 6.

2.1 How to install

2.1.1 Run5, Run6 and Run7 1st pass (LASER walks)

The code used prior to run7 is tagged `run5-offline_prod` in CVS, so the appropriate command for checking out the right version is
`cvs co -r run5-offline_prod offline/analysis/emcTimingOffsets` .

In Year 5 a macro and scripting system was used that automated the process, keeping track of ready-to-calibrate and already calibrated runs. (It's located in the `run5-work` subdirectory. Please refer to the main README file in to find out how to use this system.) From Year 6 on there was no need to use it because we used online run-by-run calibration.

Recommended environment is pro.71 for Run5. The package is to be installed the usual PHENIX way (`autogen.sh & make install`). Make sure that the place of installation is in the LD_LIBRARY_PATH.

The general root macros and shell scripts are located in the `macros` subdirectory.

Please refer to the README for a detailed description on the Year 5 scripting system and its usage.

2.1.2 Run7 repass (DATA walks recalibration)

In the Run7 offline repass we used the newest version of the code, by the time of May 2008.

Enviroment is new (by the time of May 2008. This may change, but please make sure to use newer than pro.78) for Run7. The package is to be installed the usual PHENIX way (`autogen.sh & make install`). Make sure that the place of installation is in the LD_LIBRARY_PATH.

2.2 Producing the tower t_0 offsets

One can either work from nanoDST's, or from raw eventdata (PRDF). The steps of this process:

1. Collect data files (see Sec. 2.2.1, 2.2.2)
2. Reconstruct photon ToF (using no t_0 corrections, or even using already calibrated ToF.)
3. Fit tower t_0 offsets
4. Save the output
5. Chain the output files
6. Produce the offsets (fit ToF peaks to obtain tower t_0s)
7. Reject noisy/blind towers and those with an illegal/nonsense timing
8. Test the results
9. Put the values and the reject list into the database

The class that actually does the major part of the work is `emcTofTowerOffsetReco`. Clusters are considered a photon candidate if TDC > 50 and $ADC > 20$ (in terms of low gain).

2.2.1 Tower ToF reconstruction from nDST

Reconstruction is governed by the root macro TofTowerOffsetReco.C. Interface:

```
TofTowerOffsetReco(const char * inFileList,
                   const char * outdir = ''.'')
```

Now there are more macros for basically the same task.

macros/TofTowerOffsetReco.C was used eg. in Run5. The inFileList here refers to a file that should be the following format: Each line should contain a DST-quadruplet corresponding to the same segment, of the following flavours: `DST_EMCTwr:PWG:CNT:DST_EVE`. As DST_EVE became obsolete and unneeded now, an easy workaround is to replace it by any other flavour. Or, if one desires a nicer solution, the macro is easy to modify.

Note: `run5-work/files/FindFiles4` is a handy tool of collecting dst's of demanded runs and making lists of singlets, doublets, triplets and quadruplets of different flavour dst's for each segments (see the README for details).

run7-work/TofTowerOffsetReco.C, used in Run7 repass, has been changed so that it uses list of DST files of a single flavor. This macro reconstructs collects ToF values from (already calibrated) emcClusterContent::tof(), by enabling `emcTofTowerOffsetReco::setWorkFromClusters(true)`. When working on CWG flavour,
`emcTofTowerOffsetReco::setClusterNodeName("PhPhotonList")` should also be called.
For PWG it should be the left to be the default `"emcClusterContainer"`

run7-work/TofTowerOffsetReco_WalkRecal.C is the same as above with the only difference that it envokes a module for walk recalibration, in order to go from LASER walks to DATA walks. (See Sec.4 about slewing, and the documentation on the `emcTofWalkRecal` class [3].)

Either of the above be used, it is recommended to split up the input file list into several files and parallelize the reconstruction. The Condor job file `mtowers.job` and the corresponding `mtowers.csh` script supports submission of several jobs within a single cluster. The output will then be named
`outdir/moutput.$(PROCESS)/emcTofTowerOffsetReco.root`.

2.2.2 Tower ToF reconstruction from PRDF

Reconstruction is governed by the root macro `RawTofTowerOffsetReco.C`. Interface:

```
RawTofTowerOffsetReco(const char * inFileList,
                      const char * outdir = ''.'')
```

The inFileList parameter refers to a single PRDF file (!).

The Condor job file `rtowers.job` and the corresponding `rtowers.csh` script supports submission of several jobs of subsequent segments within a single cluster. The output will be named
`outdir/routput.$(PROCESS)/emcTofTowerOffsetReco.root`.

This is a significantly slower process than the above one, as Fun4All needs to reconstruct the event from the raw packets.

2.2.3 Producing tower offset constants

First of all, if needed, one should add up the output files of the reconstruction process eg.

```
hadd moutput.???/emcTofTowerOffsetReco.root
     moutput.total/emcTofTowerOffsetReco.root
```

The offsets themselves can be created with the TofTowerOffsetProduce.C macro. Interface:

```
TofTowerOffsetProduce(const char * input,
                      const char * outdir = ''.'')
```

where the input file is just the output of the recontstruction/chaining, eg.

`moutput.total/emcTofTowerOffsetReco.root`

.
Steps of finding the peak:

- The tower t_0 peak is fitted with χ^2-minimized asymetric Gaussian of flexible range. (In Run7 repass there was an abundance of statistics, therefore it could be simplified to a "traditional" Gaussian fit on a narrow range. See documentation on the `emcTofFitter::gausfittwice(...)` and `emcTofFitter::simplegausfittwice(...)` methods)
- Towers are considered bad in "ToF" if the fit fails (either because there are less than 20 hits in them within the range, or if $\chi^2/nDF > 5$)
- The noisy towers are filtered out on request if any of the 5 predefined energy ranges contain at least 3 times higher number of hits than the average (and then the process is repeated as many times as it is needed). This function is activated/deactivated by the `void setFilterNoisyTowers(bool)` method.

The following outputs will be created:

- `outdir/ToF/*-TOWERS-T0-DRIFT.TofT0s` (ASCII tower t_0 constants)
- `outdir/RejectList` (Bad/noisy towers list)
- `outdir/emcTofTowerOffsetProduce.root` (root file for result analysis)

2.2.4 Test of the output

The `emcTofPWGHisto` class is the tool for it. Looping over emcClusterContainers of PWG files and produces root files with a THMulf, organized in subdirectories by runnumber.

macros/TofPWGCheck.C will read calibrations from DB by default, unless a different source is specified in the macro body.
Usage: `TofPWGCheck(const char* inFileList, const char* outputdir)`
`inFileList` is a list of PWG files, the output files will be under `outputdir`. In Run7 repass some altered versions of this were used.

- **run7-work/TofPWGHisto.C** also loads an `emcTofWalkRecal` module to recalibrate for t_0, but will not recalibrate for walk. Will use ASCII for recalibration.
- **run7-work/TofPWGHisto_WalkRecal.C** loads an `emcTofWalkRecal` module to recalibrate both t_0 and walk. Will use ASCII for recalibration.
- **run7-work/TofPWGHisto_recaltest.C** uses the MasterRecalibrator before the reconstruction.

Usage: `TofPWGHisto[_XXX](const char* inFileList, const char* inputdir, const char* outdir)`, here `inputdir` specifies the location of the ASCII calibration constants (TofT0Bs and, if required, TofWalks). Note: If there is walk recalibration, such a T0 set must be used that was obtained with walk recalibration too.

Macros for obtaining sector-by-sector plots, with the $ecent/e > 0.8$ photon selection by default, are in `run7-work/sectors/` Use `onerun.C(char * runnumber)` to do it for the output of one run. Or, the `doall` shellscript does all that it can find at once.

To get overall statistics: chain the output root files (`hadd pwgadded.root pwg_*.root`) and execute `runbyrun.C()`.

2.2.5 Writing constants into the Database

By default the macro initializes emcTofTowerOffsetReco with the destination set to emcManageable::kFile_ASCII. In the case one changes this to emcManageable::kDB_Pg, the tower t_0 constants and the list of rejected towers will directly go into the DB. However this is not the encouraged way, as it is recommended to check that the results make sense.

By default the run-by-run sector offset production reads the PHENIX production database. There is a way to make it read ASCII files, but at one point we need everything in DB anyway. There is a tool designed for putting the ASCII to the DB—one can do it the following way:

- Look up beginning and end validity from run control log
- Put into DB using emcDB command, eg.:
  ```
  emcDB --update outdir/ToF --forceDate "2007,01,01,00,00,00" \
  --forceEndDate "2007,08,01,00,00,00"
  ```

NOTE: `infty` as an end date should not be used anymore. The EMCal group have agreed to use closed time intervals (ie. always to specify the end date of the validity period) in order to avoid confusing already running codes with a DB update.

2.2.6 Manipulating constants

There can be a need of altering the constants before they're written into the database. Some macros are in the `run7-work/dbupdate` (and also in the `trash`) directory. The followings were used in the Run7 repass.

- `dbupdate_towert0s.C(const char* dir,const char* destdir, bool addup, bool shiftsecs, bool write)`
  `dir`: ASCII source of t_0's; `destdir`: ASCII target dir; `addup`: whether to add constants up with those already in DB; `shiftsecs`: whether to use the (wired-in-the-macro) shift constants; `write`: whether to save output.
  The t_0 constants had to be shifted with some global constants in the order of magnitude of 4ns for each sector. Therefore we used `dbupdate_towert0s.C("outdir",".",false,true,true)`
- `dbupdate_rejectlist.C(const char* dir, bool write)`
  `dir`: ASCII source of RejectList; `write`: whether to save output.
  Adds up RejectList with that in the DB.
- `unite_walktofs(const char* dir, const char *destdir, bool write)`
  `dir`: ASCII source of new walks; ASCII destination of united walks; `write`: whether to save output.
  "Unites" two walk sets into one so that the ASCII (new) set will be value1, and the DB (old) set will be value0 in the united set. This implies that the reconstruction will use the new set, but the recalibrator will be able to reach both.

2.3 Sector t_0 Tracing Offline

ToFconstants for each sector are determined on a run-by-run base. There are two Fun4All modules that need to be applied in sequence:

1. The ToF with the existing tower t_0 constants are recalibrated by the `emcTofSectorOffsetReco` class
2. The photon ToFs are collected for each sector, the peak offsets (sector t_0 values) are determined and written out by the `emcTofSectorOffsetProduce` class

The root macro `TofSectorOffsetReco.C` is meant to produce the sector offsets for a particular run.
Interface:

```
void TofSectorOffsetReco(const char* inFileList,
                         const char* inputdir,
                         const char* outdir=''.'')
```

where `inFileList` should consist of flavour-quadruplet lines from the same run, `inputdir` is the a directory containing the ASCII tower t_0 files (this is not relevant if reading from DB), and `outdir` is the directory where the sector t_0 output goes.

One can set the source and the destination to DB or ASCII by setting the `emcManageable::EStorage towert0source` and
`emcManageable::EStorage sectoroffsetdestination` variables to the
`emcManageable::kDB_Pg` or `emcManageable::kFile_ASCII` values respectively.

3 The Online Sector t_0 Calibrator

This piece of code is a module of the standard PHENIX online calibration system [5] maintained by C. Pinkenburg and being started by the shift crew for every accepted physics run. The basic idea behind it is the same as the offline version, and the core of the code is similar. The main difference is that instead of the sequential recalibration and reconstruction, all the and analysis is done in a single module that runs over raw (PRDF) eventdata. Standard Fun4All EMCal reconstruction module (EmcReco3) is used [4] to obtain clusters. Presence of the PPG and pass-0 physics calibrations for that particular run are required.

3.1 Installation and Users' Guide

This is the standalone mode only for testing, not the regular way to execute the code.

1. Set up the online environment.
2. Check out the `online/calibration/onlcal/` module [6] from CVS and use the `macros-scripts/install_onlcal.` script to install it.
   Another way is to check out the `online/calibration/onlcal/macros-scripts`, and, if you want to modify the code, online/calibration/onlcal/subsystems/emctiming" modules.
3. Start `root -b -q 'OnCal_EmcTimingCal.C()'` (If complains about environment variables: set them (you'll need a data PRDFF list file), then retry. If status=SUCCESS: Find the output root file xxx.root, then launch `root -l 'Draw_EmcTimingCal.C(''xxx.root'')'` By default we don't write to the DB. If you wish so, rerun with
   `'root -b -q OnCal_EmcTimingCal.C(0,1)'`

Note: It is also possible to make the code read the tower-t0 offsets from ASCII: one has to put the desired ToF/ dir into your workdir, then just change the corresponding line in the macro to `emctime->ReadFromPdbCal(0);`

4 Slewing (Walk) Issues

The effect that the measured TDC value of the EMCal hit depends on the ADC value is called the slewing (walk) effect.

It comes from the following sources:

1. Electronics: in case of signals of the same shape but different energy, the discriminator level is reached at different times.
2. Geometry: hits of particles with different energies induce a shower at different depths in the tower, thus the light reaches the PMT at different times again
3. Other, unresolved reasons

Slewing (walk) is usually assumed to follow a reverse power-law shape and required to saturate.

$$\text{walk} = \frac{k}{(E - E_0)^\alpha}$$

(E is transformed into ADC units. One usually neglects the E_0 as it brings another parameter to the fit. However, with enough real data it could, and probably should, be involved) It should be noted that a log shape works pretty well for the lower energy regions, and this is what the PbGl uses.

As mentioned above, timing calibration relies a lot on the slewing correction.

4.1 Laser vs. Data

For the PbSc, a standard way is to obtain these coefficients from variable-energy laser events. Every year one (or more) run is taken while no collisions, having the laser sweep all its intensity range. This has significant flaws (See Fig. 3 and 4):

1. The signal from a laser "hit" is not necessarily the same as a real photon hit

2. By a laser event, all towers fire at the same time, which is a whole lot different situation for the electronics from a usual, low-multiplicity physics event

3. The laser intensity is limited in several channels (ADC can go up no higher than 1000 in some cases)

However, for sure, one definite advantage it has: One can collect plenty of statistics in almost no time.

This latter point is a huge drawback from making walk corrections from real data. One has to collect enough hits in each channel for the higher energy region as well—requiring many runs rich in photon hits, to fit them with the desired function form. (This is usually done by the `phnxemc` user with the `emcRTM` program. The usage is the very same as that of `emcTM` [7], the only difference is that this executable provides a ROOT prompt after the analysis done.) As slewing seems not to be stable, one needs the correction coefficients as soon as possible, just to be ready to start with the timing calibration process. In case of heavy ion collisions it means several days of data taking, while in case of p+p it can mean a significant part of the whole period.

4.2 Extracting the Slewing from Data

In the case of Year 7, a Fun4All package called `emcscanraw` [8] was used for this purpose. It consists of a single class of the same name that does all the work. Steps of computing slewing from:

1. Check out the module from CVS, build and install it the usual way.

2. Go to the `work` subdirectory

3. Loop over lots of data. Go to the `work` subdirectory and use root to launch

   - `root -q -b run_prdf.C(0,"example.PRDFF","scan_out.root")`
     to process raw data (a single segment at a time, `example.PRDFF` here).
   - `root -q -b run_dst.C(0,"pwglist","scan_out.root")`
     to loop over nanoDST files of PWG flavor. Pwglist ought to be ascii format, containing one segment of root file each line
     (like `PWG_productiontag_runnumber-segment.root`)

   It is recommended that data processing be parallelized, writing differently named output files of course. The output itself will contain unevenly binned walk histograms (TDC + t_{BBC0}/LC vs. *ADC* value) for every PbSc tower (by default), and associated number-of-hits histograms— `TH1 * tof_b_`*#tower* and `TH1 * nhits_`*#tower* respectively.

4. The root macro that chains the output files, `havg.C()`, is a specialised modification of the good old `hadd.C()`. The main difference that it calculates the average of the walk output histoes with proper weighting for each-and-every bin. One has to hand-edit the `havg()` function in order to include the proper list of root files. Then just run `root -q -b havg.C`. Its output has the default name `scan_added.root`.

5. There are a couple of tools that can be applied on these root files. One can store and open any stage of a process. First, one needs to load the macros from root: `.L walkutils.C` , then:

   - `void extractwalk(runno, "input.root","output.root")` : Extract the walk coefficients (original, so laser for Year 7) from DB, fits the new ones and stores them as `TF1 * wk_f1_`*#tower* and `TF1 * wk_f2_`*#tower* respectively.
   - `void makestats(runno, "input.root","output.root")` : This one is to create the general distribution histograms shown in Sec. 4.1. If the TF1's are not present in the input, the above process will be invoked.
   - `void updatewalk(runno, "input.root")` This one exports the new fitted walk coefficients into DB format (ASCII files under the ToF directory, by default) If the TF1's are not present in the input, the above process will be invoked.

6. Some channels usually cannot be fit due to lack of statistics or other reasons, and the corresponding lines in the ASCII files will contain a *nan* for the walk value. Besides of causing an IO problem, we lose valuable statistics, so instead of flagging these towers as bad/warn, there is a script to fix these

failed fits: `updfail.sh` replaces these with the calculated average walk of the corresponding individual supermodule. (The script looks for input in `./ToF`. One probably wants to replace the upper and lower boundary for values accepted to get in to the average.)

The following macros (in the same `work` directory)were used to produce the plots for this analysis:

- `tdcadc.C(channel_no, "input.root")` to create the TDC(ADC) plots (fig. 3)
- `profiles.C("input.root", colorscheme)` to create the geometrical distribution plots of the laser walk. The `colorscheme` switch can take 0 for continouos, 3 for 3-color and 4 for 4-color scale. (Fig. 5)

4.3 Comparison and conclusions

One can find that the walks from data are much more reliable. The distribution of the laser slewing coefficients on Fig. 4 clearly shows a nonphysical tail, and if one looks at the geometrical distribution of the constants on Fig. 5, it becomes clear that the problem corresponds to some FEM's and ASIC's. We found the following FEM's to behave suspiciously: #42,#49,#60,#61,#68,#72,#97)

Having a comperative look at the timing peaks from both sets of walk in Run7 (Fig. 8; Sec. 6.3) tells us that, while the peak widths do not differ significantly, the data walks give us a better shaped distribution with more photons timed well. We can conclude that ToF with walks from data is more reliable, and definitly this is that should be used for ID purposes.

5 ToF and Walk Recalibrator (Run7)

EmcTofWalkRecalReco is a replacement for the [...]Emctofrecalreco recalibrator classes in run7, and it is based on standard calibration methods.

5.1 Usage

The corresponding class is `EmcTofWalkRecalReco` (derived from `Recalibrator` class, that's a child of `SubsysReco`), and is envoked for Run7 runs by the `MasterRecalibratorManager`. If you need to tweak it, check out `offline/framework/recal` [9], modify, make and install it. In the case you do not want the other recalibrators to load, you can instantiate and register this module in your macro as a traditional `SubsysReco` as well.

5.2 Purpose

There are three different functions the module can fulfil:

- "Direct" recalibration: It will use the newest valid production Pg values and the raw ADC & TDC. Warning: because of its need for raw values, it can't work with older versions than emcClusterContainerv5; and so will it crash on a PhPhotonList! If there is no raw ADC value in the container (that is present from v6), or if demanded, the ADC value is restored from the gains.
- "Afterburn": Will UNDO old calibrations (source specified in the `InitRun(...)` as second parameters of the `setSourceXXX(emcManageable,emcManageable)`. Substitute XXX for TofT0bs, WalkTofs and SectorTofs respectively) and then REDO it with a new set of calibrations (source specified as the first parameters of the above methods). In principle it'll work on any emcClusterContainer that has the ToF fields.
- "Single-Afterburn": Same as above for the *walk*, but with only one calibration dataset. When "WalkTofs" calFEM's are read, the old walk constants (for undo) are taken from the previously unused value0, and the new walk constants (for redo) are from the usually used "WalkTofs" value1. The t_0 is afterburned by using the calFEM "TofT0b" values as an additive correctional factor (hence no t_0 undo).

By this time (May '08) the choice is hardwired to be "Single-Afterburn" in the `InitRun` method, as this is the one suitable for the Run7 afterburning. It can be changed to depend on runnumber if required. Note that both the "afterburn" and the "single-afterburn" methods rely on the ToF values saved in the `emcClusterContents` and therefore a new t_0 will be required if they change (eg. in case of a new production).

Remark: `emcTofWalkReco` (that of the offline Tof calibration, Sec. 2.2.1) is of the very same purpose and uses the very same method as the recalibrator when the "Afterburn" function is selected. That is the one that was used for producing the Run7 recalibration offsets.

6 Calibrations Applied to Data

In the PbGl case, the theoretical resolution of ≈ 550 ns of the electronics can be reached, while one could never get near to the physical limits of ≈ 150 ns of the PbSc towers (Table 1). Theories are plenty in number, but unfortunately there is no solution for it so far.

Table 1: Resolutions for different periods of operation.

Year/Species	PbSc Δt (ns)	PbGl Δt (ns)
Year 1 Au+Au	≈ 700	$550-600$
Year 4 Au+Au	$350-450$	$550-600$
Year 5 Cu+Cu	$500-600$	$550-600$
Year 5-6 p+p	≈ 800 (recalibrator)	$550-600$
Year 7 Au+Au	$400-500$ (single run offline†)	$550-600$

† Determined after producing a tower $t_0$set from a single run #239461 offline PWG files, after data walk recalibration was applied.

6.1 Year 5 Cu+Cu

The same method was applied as by L. Aphecetche in Year 4 Au+Au. Slewing deffinition of walk = 4000/ADC was used, with coefficients derived from laser data. Around $500-600$ ns for PbSc and PbGl was achieved (ie. poorer resolution than the Year 4 $350-450$ ns for the PbSc; See Fig. 6).

6.2 Year 5 and 6 p+p

As a preparatory step for the 500 GeV data taking, the gain definition changed between the two different species of Year 5. This effected our timing calibration. For Year 6, an online calibration was applied. With this, around $550-600$ ns for the PbGl could be achieved. However, as we found out, the PbSc laser data that was taken for the slewing correction was taken in the wrong intensity range and the fits went unstable, causing the walk coefficients to be completely unusable. Therefore we had to apply a "patched" sector-by-sector walk correction, that was barely enough to achieve around $800-1000$ ns in the PbSc.(See Fig. 7)

After Year 6, the laser runs were repeated (run 216817). Several attempts were made to find a definition that describes laser data better. For small energies, the best fit was still given by a logarithmic shape, but it became impossible to extrapolate for the higher energies [10] (Fig. 2). Therefore the EMCal group agreed to use walk $= wk * 1000/\sqrt[3]{\mathrm{ADC}}$ for the PbSc.

6.3 Year 7 AuAu

For the 1rst (online) pass, tower t_0 sets were produced using laser run 224131 with the above inverse cubic root walk definition for the PbSc, and then online calibration was applied to obtain sector t_0's. After the online production QA made it clear that PbSc timing resolution is not acceptable ($800-1000$ ns again) [12]. So we extracted the slewing coefficients from online produced nDST files. Significant differences between the two set of slewing coefficients can be seen (Fig. 4).

In order to get a more acceptable timing resolution, the following process was applied to the PbSc only: We looped over 100 MinBias PWG pro78 segments twice, without recalibrators, in order to produce new sets of tower t_0's.

1. For the 1st set of tower-t_0 values: the original (laser) slewing constants was used
2. For the 2nd set of tower-t_0 values: first of all, we subtracted the effect of the timeshift made by (laser) slewing in the production, then calculated the new timeshift from the data walk. The new (data) slewing constants were used when collecting the hits (see Sec. 2.2.1 for details).

Then, using the above tower-t_0 sets, a reconstruction module was executed (with walk recalibration in the 2nd case) and ToF peaks from a (usual) tight photon selection were determined. This gave us the resolution of 400-500 ps. No significant difference is seen between the sector-by-sector ToF distribution widths of the two methods. However it is to be noticed that the peaks are around 10% higher in the case the values were recalibrated with the data walks. Also there are much less hits that outlye from the peak region, and the distribution is more symmetric, more Gaussian-like. See Fig. 8 and Fig. 9.

A summary G/H PWG talk of the Run7 ToF and walk calibration is in [13].

Timing calibration for Year 5 and 6 p+p has been implemented by K. Okada as a separate recalibrator, based on reversing the wrong calibration and applying the new ones. Latters are obtained by using all the data of the period to determine the overall tower t_0 set and five different sector t_0 sets for different regimes of runs [11].

References

[1] `http://www.phenix.bnl.gov/WWW/emcal`

[2] `http://www.phenix.bnl.gov/viewcvs/offline/analysis/emcTimingOffsets`

[3] `http://www.phenix.bnl.gov/WWW/emcal/documentation/emcal_timing_calibration`

[4] `http://www.phenix.bnl.gov/WWW/emcal/documentation/emcal_reconstruction`

[5] `http://www.phenix.bnl.gov/WWW/run/07/calibrations`
`http://www.phenix.bnl.gov/WWW/run/07/calibrations/subsystems/emctiming`

[6] `http://www.phenix.bnl.gov/viewcvs/online/calibration/onlcal`

[7] `http://www.phenix.bnl.gov/WWW/emcal/computing/online/EmcOnLineDoc/EMCalMonitoring.html`

[8] `http://www.phenix.bnl.gov/viewcvs/offline/analysis/emcscanraw/`

[9] `http://www.phenix.bnl.gov/viewcvs/offline/framework/recal/`

[10] `https://www.phenix.bnl.gov/WWW/p/draft/vertesi/emcal/070122_walk/`

[11] `https://www.phenix.bnl.gov/WWW/p/draft/okada/070226/`

[12] `http://www.phenix.bnl.gov/phenix/WWW/run/04/dataprod/QA/index.php`
(select Run7Au+Au Central, EmcT0, any)

[13] `https://www.phenix.bnl.gov/WWW/p/draft/vertesi/talks/080527_EmCal_ToF/`

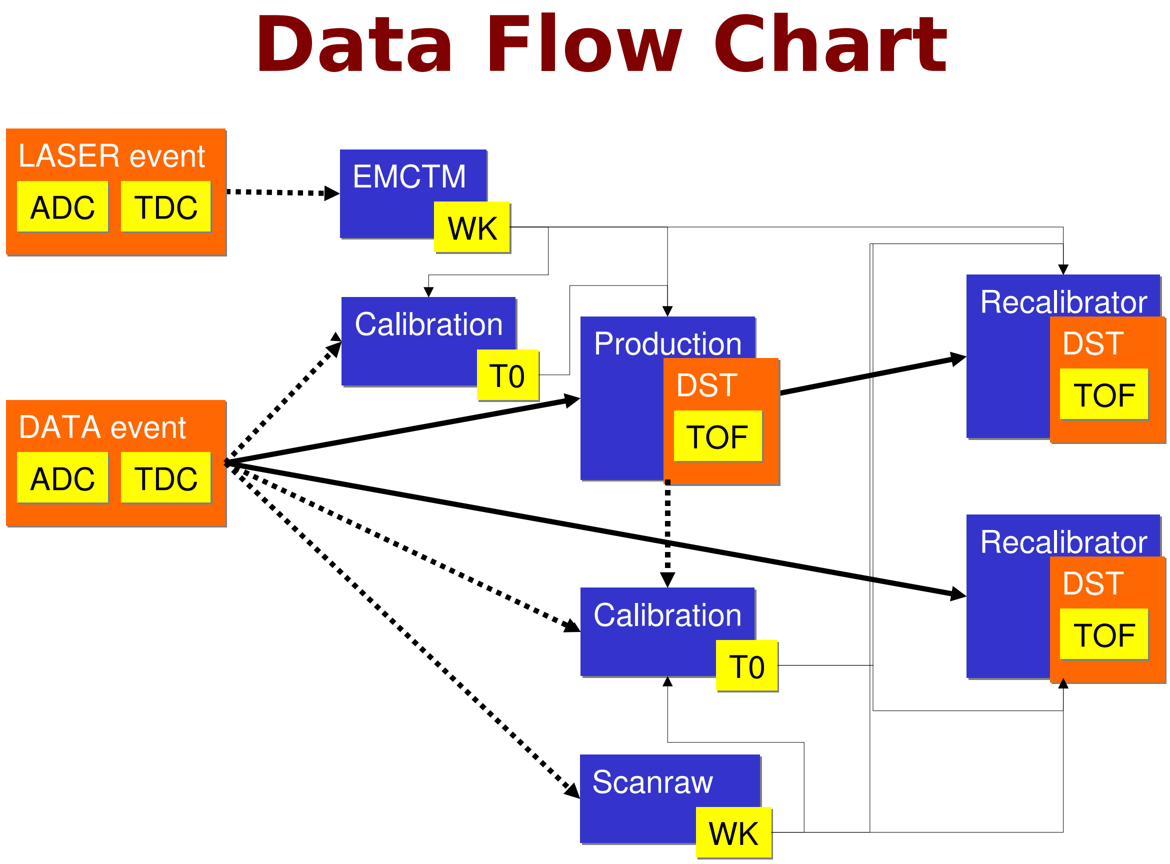

Figure 1: Data flow chart of the walk and ToF calibration

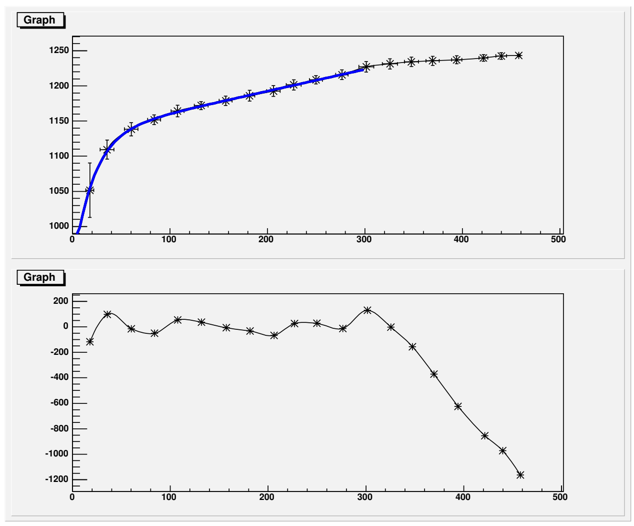

Figure 2: An example of slewing histograms from laser. The black crosses are TDC vs. ADC for laser hits. The fitted walk curves (blue) are of the shape of $c_0 + wk * 1000/\sqrt[3]{\mathrm{ADC}}$. Bottom: Difference of the laser hits and the fitted curve in the units of ps. The fit is limited to ADC < 300 as laser hits are often missing in the higher range (not here). Is obvious that extrapolation is problematic.

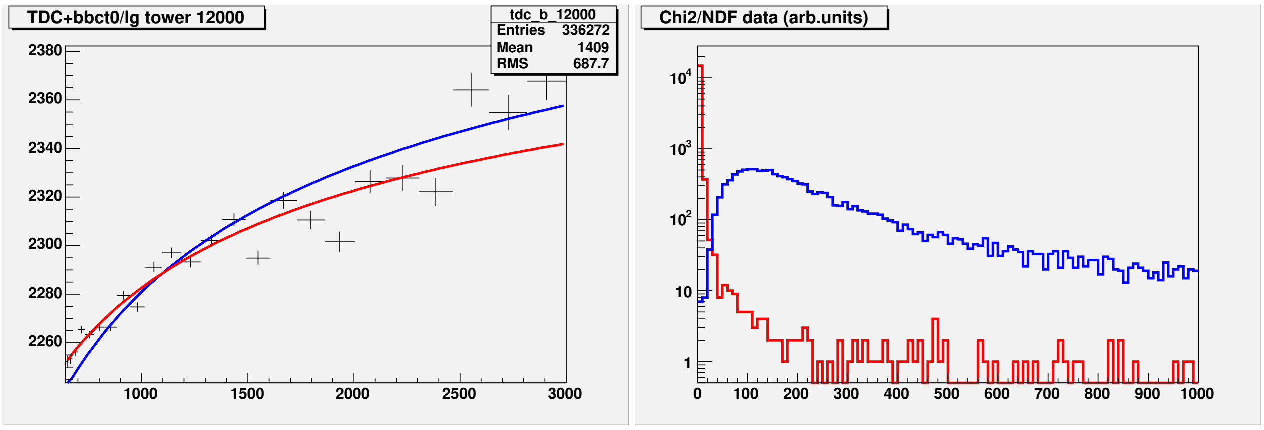

Figure 3: Left: An example of slewing histograms from data. (The black crosses are $TDC + t_{\mathrm{BBC0}}/\mathrm{LC}$ vs. ADC for selected photons; The error bars are proportional to the square root of the number of hits in each bin—to be used as inverse weights for the fits). $TDC + t_{\mathrm{BBC0}}/\mathrm{LC}$ vs. ADC from data) The fitted walk curves are of the shape of $c_0 + wk * 1000/\sqrt[3]{\mathrm{ADC}}$. In the case of the slewing coefficients from laser (blue), the wk_{LASER} values from DB and only c_0 is fitted. In the case of the data slewing coefficients, $wk = wk_{\mathrm{DATA}}$ are fitted as well as c_0. Right: The χ^2 distribution of the laser (blue) and data (red) fits for all the PbSc towers. *Note: error bars do not represent propagated errors, but are estimated from #hits in each adc bin. Therefore the absolute scale of* χ^2 *is arbitrary.*

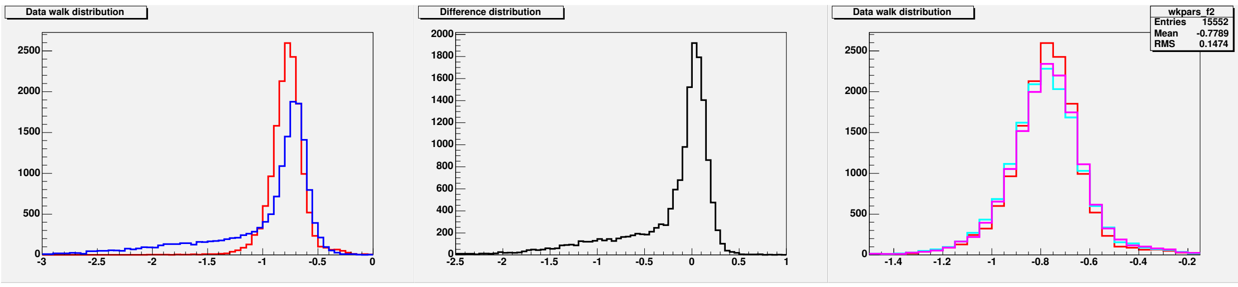

Figure 4: Left: Distribution of slewing coefficients from laser (wk_{LASER}, blue) and data (wk_{DATA}, red), for all the PbSc towers. Center: $wk_{\mathrm{LASER}} - wk_{\mathrm{DATA}}$ distribution. The peak± 0.15 region can be fitted with a $\sigma = 0.1$ Gaussian. Right: Stability of wk_{DATA} during Year 7. A Gaussian was fitted on the entire dataset (red, $mean = 0.771 \pm 0.01$, $\sigma = 0.113 \pm 0.01$), the first half of the data (cyan, $mean = 0.778 \pm 0.01$, $\sigma = 0.126 \pm 0.01$) and the remainder half (magenta, $mean = 0.770 \pm 0.01$, $\sigma = 0.126 \pm 0.01$).

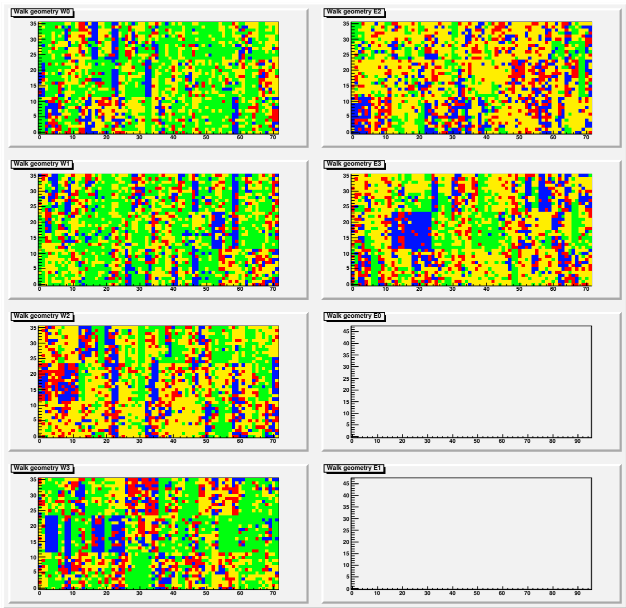

Figure 5: Geometrical distribution of the laser walk constants in the PbSc sectors (Left side, from top to bottom: W0, W1, W2, W3; Right side, from top to bottom: E2, E3). Blue: bad channel ($wk_{\rm LASER} < -1.5$); red: probably bad ($-1.5 \leq wk_{\rm LASER} < -1.0$); yellow: probably good ($-1.0 \leq wk_{\rm LASER} < -0.7$); green: good ($0.7 \leq wk_{\rm LASER} < 0$). Structure due to ASIC's and FEM's is clearly seen.

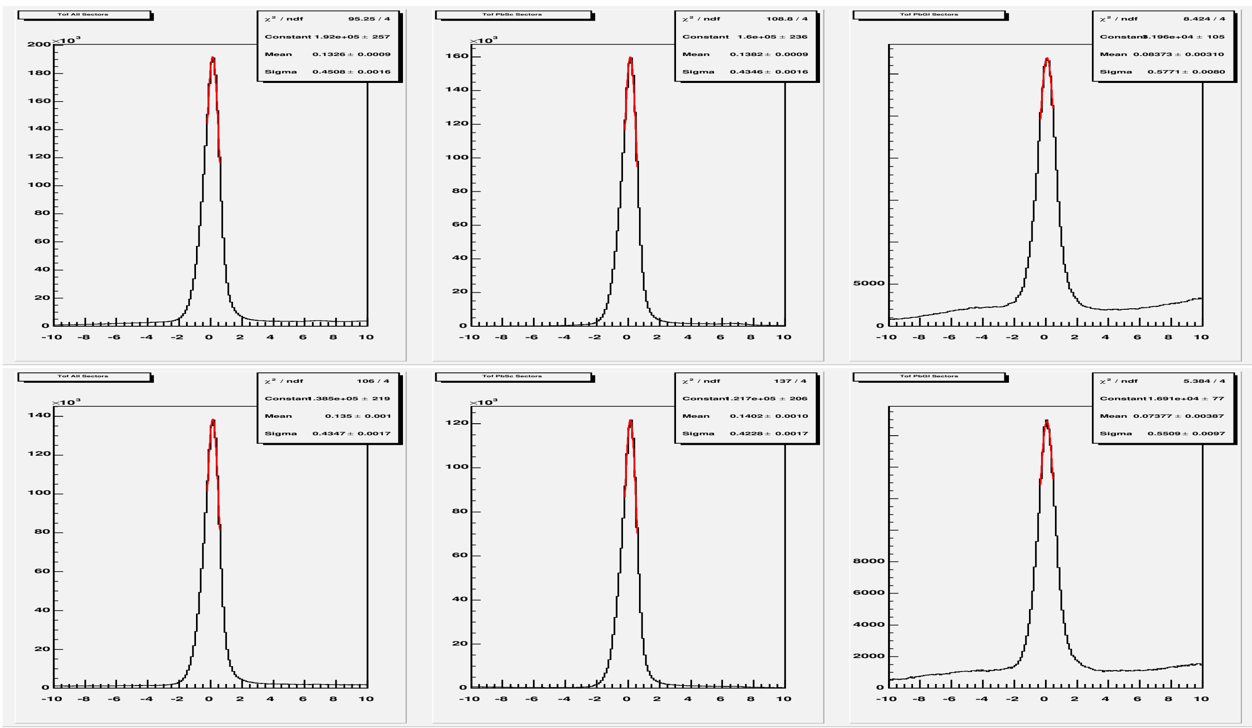

Figure 6: Year5 CuCu ToF peaks altogether and separately by PbSc and PbGl. Data samples from all the runs are shown with a strict photon selection. Top: Cu+Cu 62GeV; Bottom: Cu+Cu 200GeV.

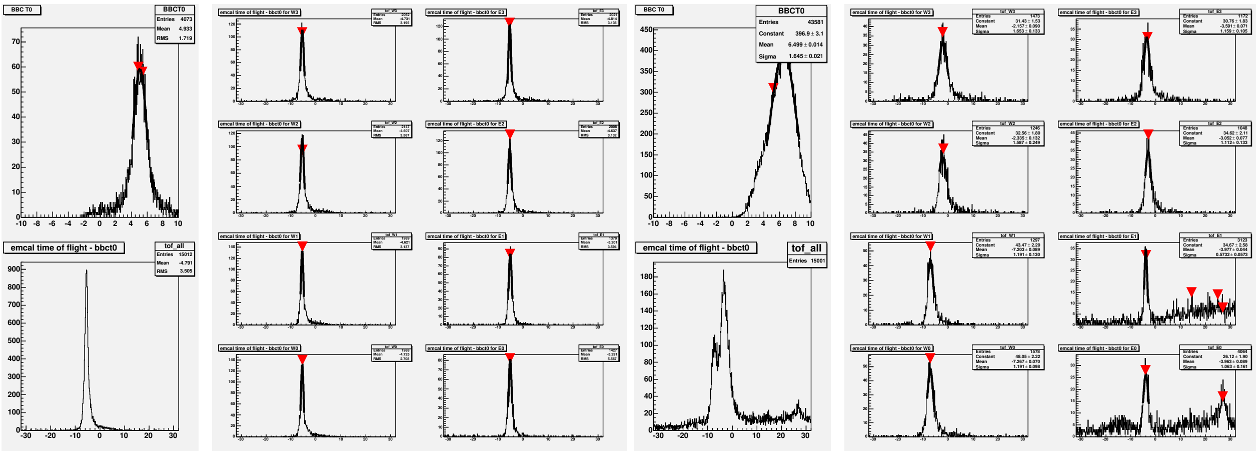

Figure 7: Online sector-by-sector calibration. Left: Tested on Year 5 CuCu (run 161598) with the expected width. Right: Year 6 pp (run 189010) with the failed walk calibration and a width of $\tilde{1}$-2ns. The latter is acceptable for a rough run-by-run sector offset alignment, but not for the final photon ToF.

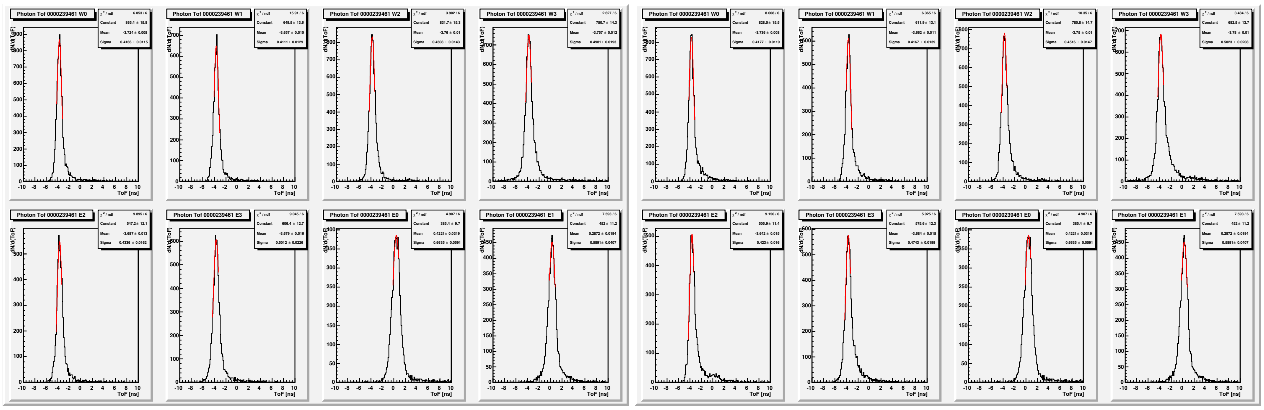

Figure 8: Recalibrated Run7 sector ToF peaks. Left: ToF from run #239461 PWG segments with DATA walk constants. Right: ToF from run #239461 PWG segments with LASER walk constants. Sectors from left-top to right-bottom: W0, W1, W2, W3, E2, E3, E0, E1.

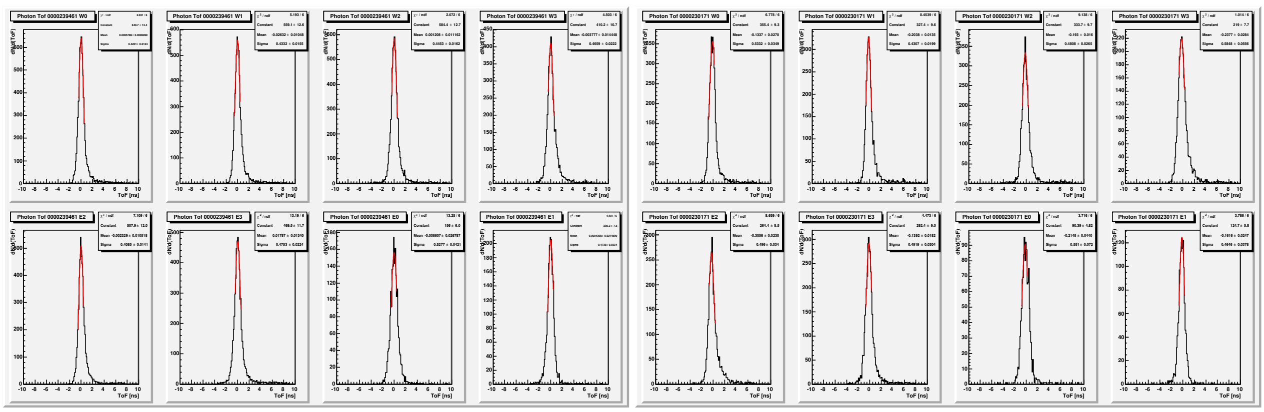

Figure 9: Final recalibrated Run7 sector ToF peaks, corrected for the sector shifts. Left: ToF from run #239461 PWG segments. Right: ToF from run #230171 CWG segments. Sectors from left-top to right-bottom: W0, W1, W2, W3, E2, E3, E0, E1